Taming The Beast

Porting Evolution Data Server To DBus

Ross Burton <ross@openedhand.com>
OpenedHand Ltd. http://o-hand.com

Who Am I?

- Ross Burton
- Employee of OpenedHand Ltd
- Main Sound Juicer developer
- Not as angry as I appear on Planet Gnome, honest

What Is Evolution Data Server?

- Front-end agnostic personal server for PIM data, storing contacts, events, tasks, memos
- Data storage separated by IPC from the client
- Simple conceptual model: Contacts, Books and Book Views
- Proven to work: Evolution, clock applet, desktop-applet, and so on

What Is Evolution Data Server?

- Has some great functionality
- Comprehensive live query functionality
- Change notification to all interested applications
- Transports data in vCard and iCal formats

What Is DBus?

- New lightweight IPC system designed for desktops
- Not designed for IPC over LAN or Internet (use REST, SOAP)
- Not designed to do everything (use CORBA)
- Many bindings make it easy to integrate into desktop applications
- High Bling Factor

Why Port EDS To DBus

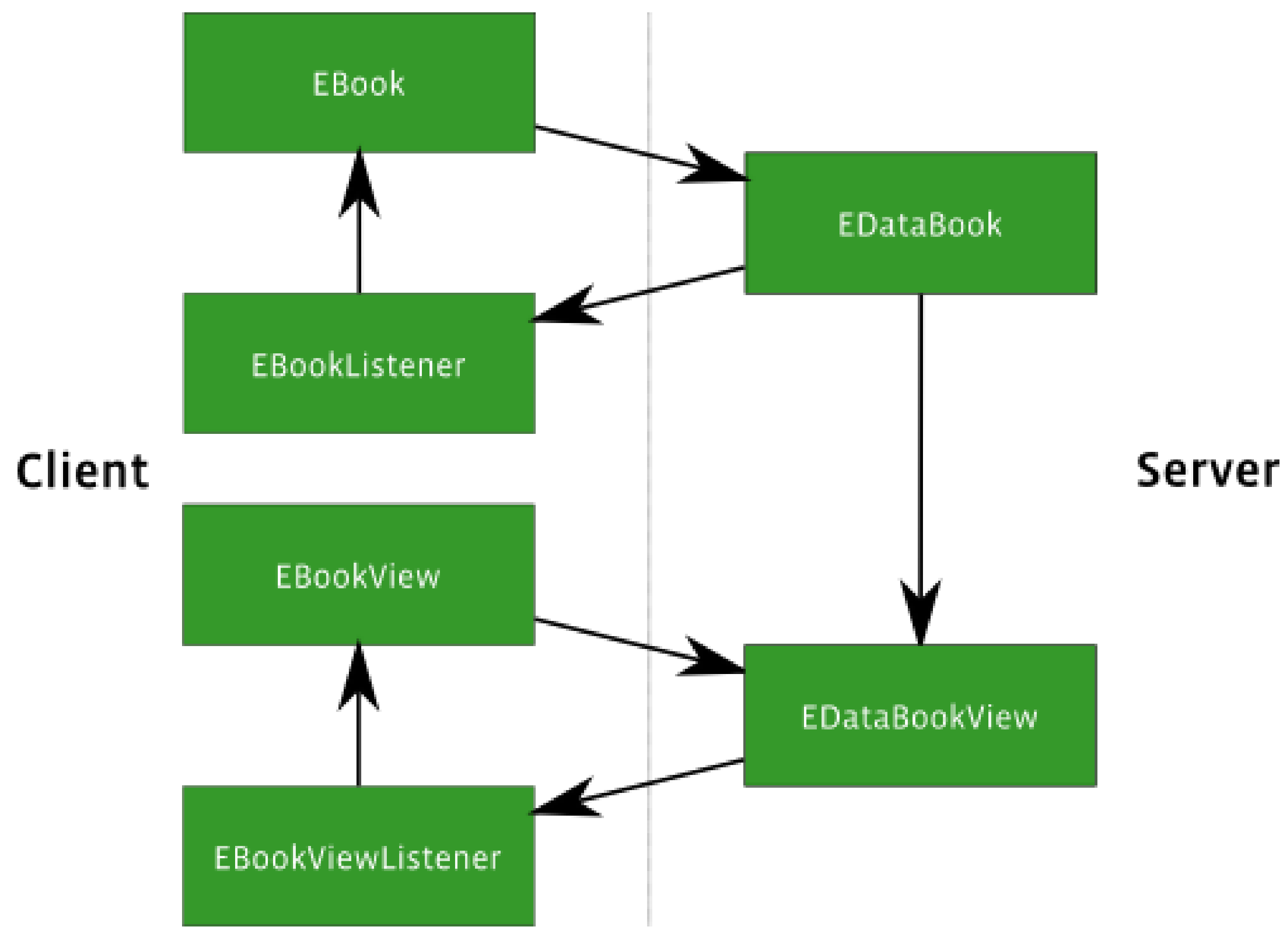
- Wanted to trial EDS on hand held devices, for decent PIM integration (thanks Nokia)
- GTK+ based handhelds only use DBus, not Bonobo (see Maemo and GPE)
- GConf and gnome-vfs have working DBus ports, so real-world ports are possible (this was 2005)

Feasibility

- CORBA overcomplicated for the requirement of EDS
- Bonobo limitations means it's non-trivial to do non-blocking method calls, need a listener object on the client and make two one-way calls
- Client library wraps Bonobo completely and provides blocking and non-blocking API

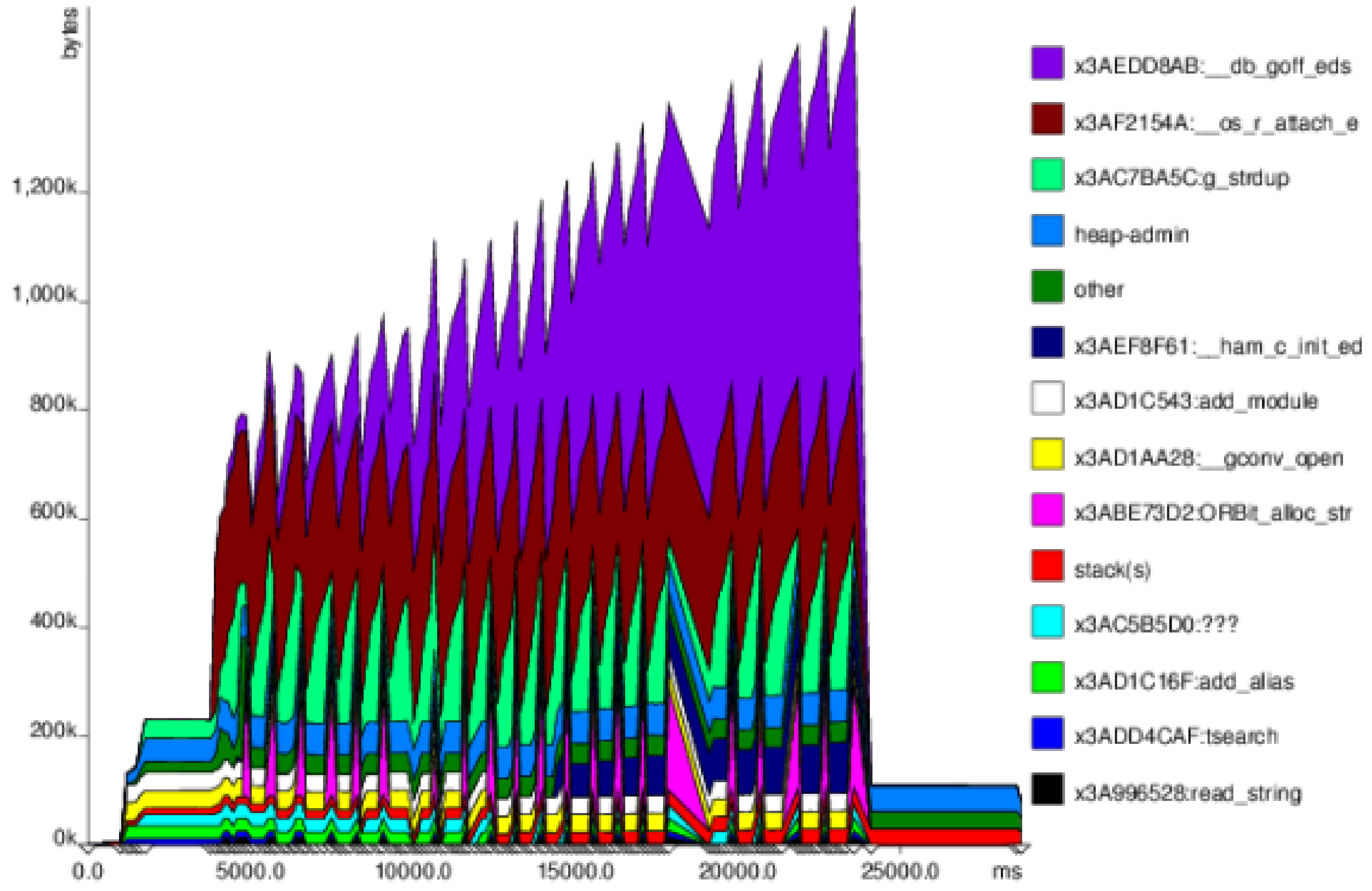
Feasibility

Over-complicated architecture



Feasibility

For reviewing memory use, Memcheck and Massif are your friends. Spot the leak:



Feasibility

- Lots of memory leaks disguising quite reasonable memory usage
- Database cursors not destroyed after every query leaked the database cache
- Threads not joined after every query causes massive VM leak
- Contacts compared for inclusion in queries even when the filter was 'select all'

DBus Advantages

- Relatively light (core library is ~200K, Bonobo is 800K)
- Solid low-level API
- Almost-great GObject bindings (some issues)

DBus Advantages

- **Easy to get a proxy to a remote object**

```
DBusGProxy *proxy;
proxy = dbus_g_proxy_new_for_name
  (connection,
   "org.gnome.Evolution",      /* Service */
   "/addressbook/BookFactory", /* Object */
   "addressbook.BookFactory",  /* Interface */
   &error);

char *path;
dbus_g_proxy_call (proxy, "getBook", &error,
                   G_TYPE_STRING, "/some/uri",
                   DBUS_TYPE_G_OBJECT_PATH,
&path,
                   G_TYPE_INVALID);
```

DBus Advantages

- IDL format is XML based, verbose but clear

```
<interface name="addressbook.Book">
 <method name="getContact">
  <annotation name="Glib.CSymbol" value="..."/>
  <arg name="uid" type="s" direction="in"/>
  <arg name="vcard" type="s" direction="out"/>
 </method>
```

- When GObject has introspection expect this to be generated from source annotations

DBus Advantages

- Simple generated GObject bindings for synchronous calls

```
GError *error = NULL;
char **vcards;

book_get_contact_list (proxy, query,
                       &vcards, &error);
```

- Call blocks until a reply is received

DBus Advantages

- Simple generated GObject bindings for asynchronous calls

```
void
callback (DBusGProxy *proxy, char **vcards,
          GError *error, gpointer user_data)
{ /* do stuff */ }

book_get_contact_list_async (proxy, query,
callback, data);
```

- When a reply is received the mainloop will execute the callback

DBus Advantages

- Binding tool generates mapping from incoming DBus method calls to C functions

```
static void
book_class_init (BookClass *klass) {
  ...
  dbus_g_object_type_install_info (
    G_TYPE_FROM_CLASS (klass),
    &dbus_glib_book_object_info);
}
```

DBus Advantages

- Synchronous server-side methods are trivial to implement

```
static gboolean
book_getContact (Book *book,
                 const char *in_uid,
                 char **out_vcard,
                 GError **error)
{
  *out_vcard = get_the_contact (in_uid);
  return TRUE;
}
```

DBus Advantages

- Asynchronous server-side methods are almost as simple

```
static void
book_getContact (Book *book,
                 const char *in_uid,
                 DBusGMethodInvocation *context)
{
  get_contact_async (in_uid, context);
}

...

dbus_g_method_return (context, vcard);
```

DBus Advantages

- Powerful object model leads to clean design
- Services, objects, interfaces, methods, signals
- Server has a factory object that creates book objects, and book objects create book view objects
- DBus has good support for asynchronous calls and signals so IPC is straightforward

DBus Limitations

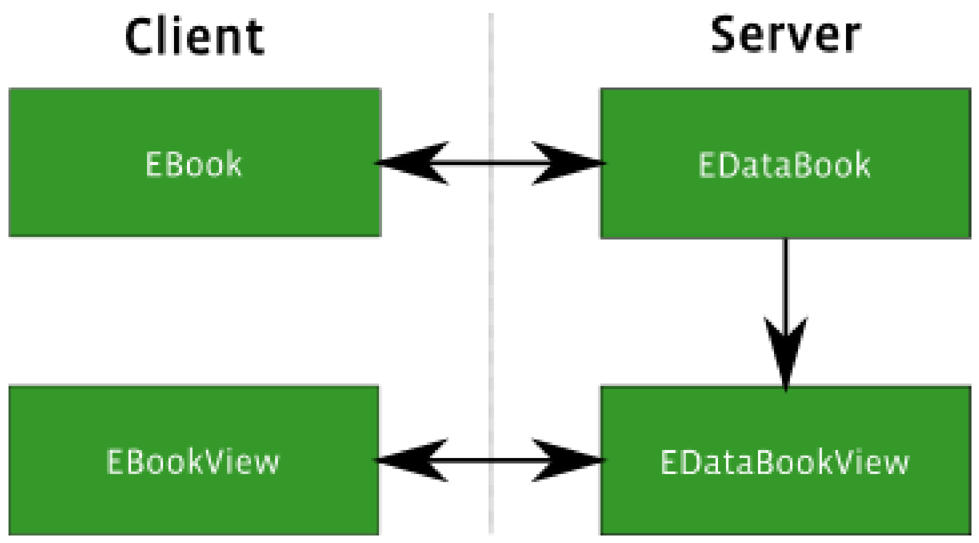
- Message copied many times as it goes through the bus
- Possible to deadlock as bindings are not re-entrant
- No automatic threading per request

Implementation

- Mostly dull — code wrapping DBus on client and calling backend functions from DBus calls on server
- Use some #defines to allow backend code to build against both DBus and Bonobo

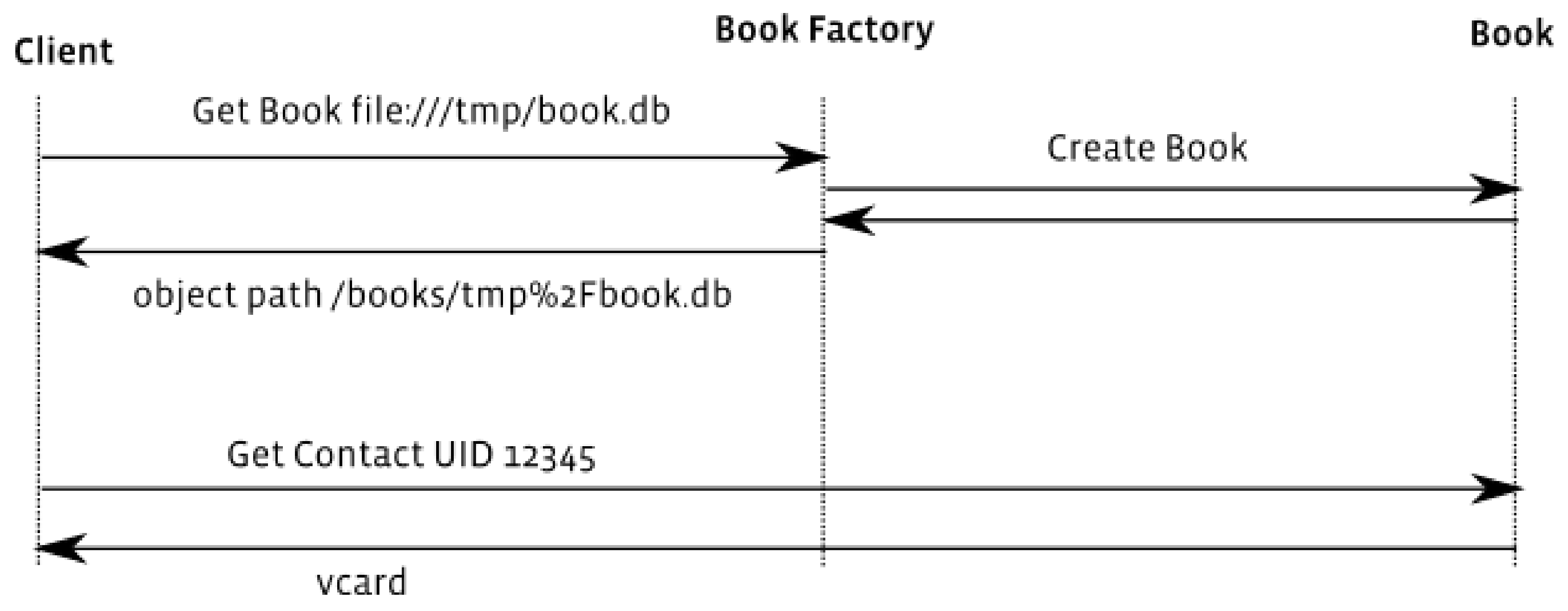
Implementation

Sanity returns



Implementation

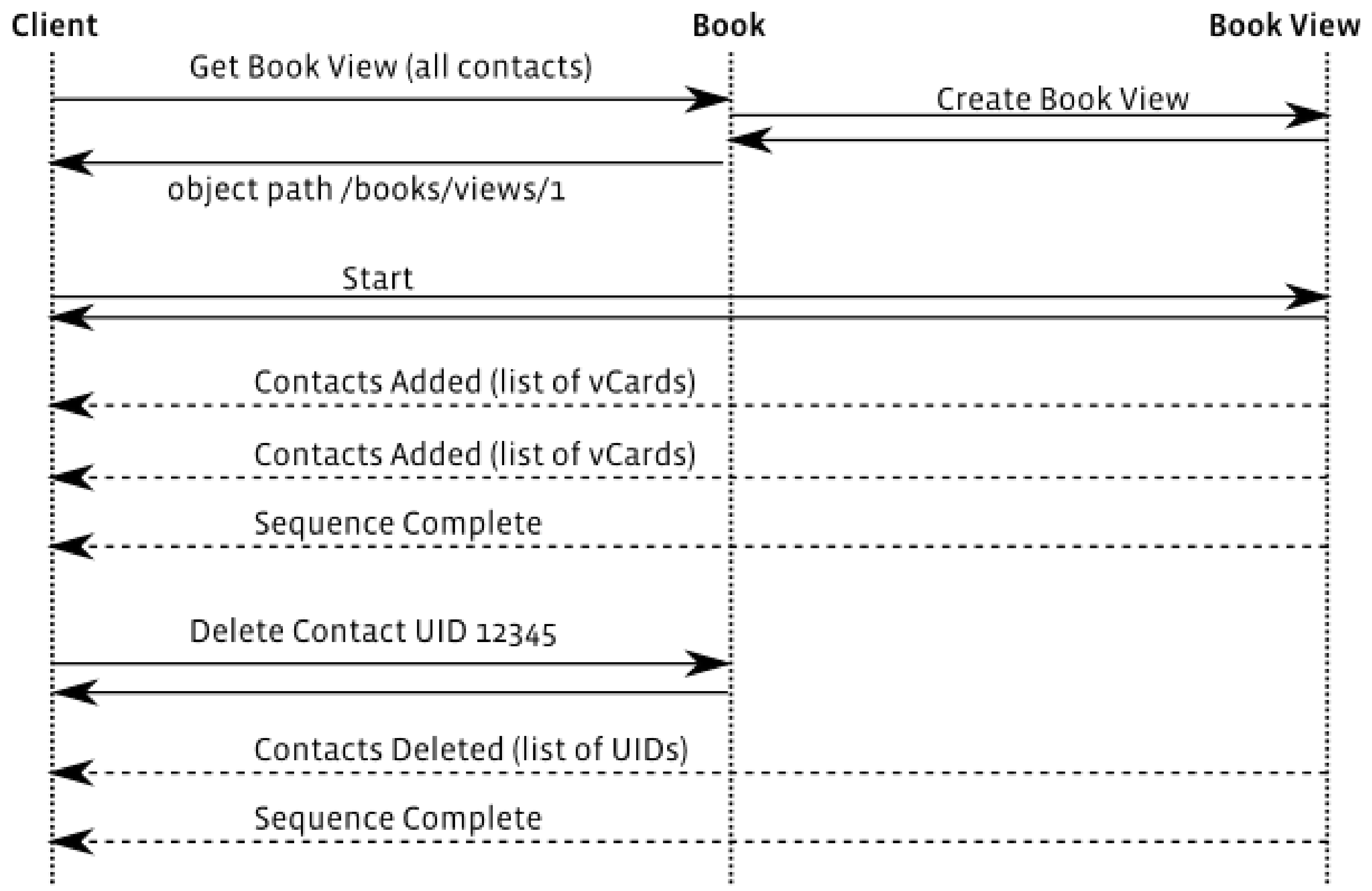
- Book Factory is a singleton which creates new Book objects when asked


Client
Book Factory
Book
Get Book file:///tmp/book.db
Create Book
object path /books/tmp%2Fbook.db
Get Contact UID 12345
vcard

- Keeps track of client connections and book lifecycle

Implementation

- Book Views are created by Books, and emit signals to notify clients about contacts


Client
Book
Book View
Get Book View (all contacts)
Create Book View
object path /books/views/1
Start
Contacts Added (list of vCards)
Contacts Added (list of vCards)
Sequence Complete
Delete Contact UID 12345
Contacts Deleted (list of UIDs)
Sequence Complete

External API Changes

- Nothing drastic
- e_book_view_new changed (but should be private)
- EContactPhoto extended
- Many new functions for performance and functionality improvements

Performance

- Reduced memory use with no known leaks
- Generally same speed or faster
- However Bonobo is faster than DBus at raw message throughput

Future Work

- Sync with EDS 1.6, currrently at 1.4
- Merge upstream, for G2.16 hopefully
- <harish> ross: I will try to get that reviewed and absorbed [...] before your GUADEC talk

Future Work

- Calendar port needs to be stress tested
- Run data-intensive book views over a private connection? Shared memory transport?
- Need some threading magic a lá Bonobo
- Port remaining backends

Demos

- EDS/DBus with Evolution
- Contacts
- Nokia 770, IT2006 Release

Thank You For Listening

- Any questions?